Tagger についての一考察

著者	名倉 秀人
著者別名	NAKURA Hideto
雑誌名	東洋大学大学院紀要
巻	53
ページ	225-239
発行年	2016
URL	http://id.nii.ac.jp/1060/00008786/

Taggerについての一考察

文学研究科英文学専攻博士後期課程満期退学
名倉　秀人

1. 二種類のコーパス

コーパスにはテキストだけのものと、各単語に文法tagが付けられたものがある。プレーンテキストのコーパスも十分に語学研究に役立つが、tag付きコーパスは、全く別の角度から語学的考察を行うことができる。どちらが優秀かという問題ではなく、目的によって使い分ければ様々な研究ができる。tag付きコーパスは基本的に一単語にひとつのtagが付いているため、単純計算でデータベースのファイルの大きさが2倍付近になる。大きなファイルになるとコンピュータがハングアウトしてしまうことも出てくる。また、コンコーダンサーの表示設定を変えなければいけないので、切り替えに手間がかかる。できれば同じテキストでtag付きのものとそうでないものを用意していれば、使い勝手が良くなるだろう。

かつては市販のコーパスに頼っていたが、今ではインターネットやコンピュータの発達により、研究者が個人でコーパスを作成しているケースが多くなっている。たいていは文学作品やニュースなどのソースからテキストを抽出してコピー &ペーストしたテキストファイルだ。コンコーダンサーも発達して、AntConcはひとつのテキストファイルだけではなく、フォルダーに入った複数のテキストファイルを検索にかけることができるので、ファイル名にそのソースを記して複数ファイルで収集する方法もある。プレーンテキストでのコーパス構築術は十分に発達したと思う。

一方、tag付きコーパスの作成をしている学者はさほど多くはない。理由はいくつかある。まず、tag付きコーパスの意義があまり見いだせない。プレーンコーパスだけでも十分分析ができるように思える。また、tagの付け方がわからないということもある。taggerの存在は知っているが、使用法がいかにも理系的で難しそうに思える。さらに、tag付きコーパスをどう使うかがわかり難いのも事実だ。ある文法的構造をした例文を抽出したい場合、コンコーダンサーでのコマンドの出し方がプログラムのようで戸惑う。British National Corpusはtag付きコーパスだが、tagを使わずプレーンテキストのコーパスとしてのみ使う学者もいる。このように、tag付きコーパスには難しいというイメージが付いてしまっている。しかし、ようやく最近になってtaggerが非常に使いやすくなってきた。Windowsにも対応し、それ

に合わせてコンコーダンサーも徐々にGUI対応になってきている。このtaggerというものを、少しだけわかりやすくしようというのが本稿の狙いである。

2. Taggerの歴史

かつては文法を学ぶ学者や学生が手打ちでtagを付与していたこともあった。しかし、これではあまりにも効率が悪い。そこでコンピュータで一気に自動的にtagを付けるプログラムが作成された。しかし、その正確さは90%をなかなか越えなかった。90%と言えばかなりの正確さに見えるが、単語10個に対して1つのミスがあってはデータベースとしては不安である。British National Corpusの作成時、CLAWSという正確度が高いtaggerが1983年に作られた。当時でも96-97%の正確性があった。これを手に入れることができれば、tag付きコーパスを作ることができるはずだった。しかし、CLAWSはLancaster大学によって公開されることはなかった。開発したコンピュータはいわゆるパソコンではなく大型コンピュータでOSはUNIXという専門的なものであったためである。値段も高く、個人が扱えるものではなかった。その後、1993年にEric Brill氏によってBrill's Taggerというフリーのtaggerが開発された。これもUNIXベースではあったが、Linuxに対応したりDOS用にコンパイルしたものが出てきて、パソコンでも使えるようになった。GoTaggerはBrill's TaggerをWindows上で使えるようにしたGUIプログラムである。2005年に後藤一章氏によって開発された。その他、Helmut Schmid氏によって作られたTree Taggerもある。Brill's TaggerとTree Taggerは無料である。そしてついにPC版CLAWSが発売された。2010年にはJAVA版が出ていたようだ。本稿は主にCLAWSのtagsetについて調査していきたいと思う。

3. CLAWSの購入とインストール

いつごろからかは判明できなかったが、Lancaster-UniversityのサイトでCLAWSのライセンスを販売するようになっていた。数年前にWINDOWS用のCLAWSを開発しているという発表があったので、気にしてはいた。ネット上で無料で使えるCLAWSもあるが、これは文字数に制限があるため、自分の作成したデータにtaggingをするのには向いていない。スタンドアローンで扱えるCLAWSがようやく手に入る時代となった。

lancaster-universityのサイトを開くと3つのCLAWSとWmatrix というネット上で使えるコーパス分析サイトのライセンスを販売している。WmatrixにはCLAWSが含まれるのだが、ネット接続が必須なことと、1年間のライセンスであるためCLAWを選択した。3つのCLAWSは20人までの複数ライセンスと個人のためのライセンス、そしてネット上で使うライセンスがある。それぞれ、2016年6月時点で、£350.00, £100.00, £750.00の価格が示されていた。個人の研究のため、Single User Licenseを選択した。数を1にし、“Add to Basket”で購入ソフトを決定すると、“Which operating system will you be using?”と出てくるの

で“Windows 8.1”と書いてnextボタンを押すと購入確認のページに値段が出てくる。100ポンドなので、この時点で約1万5千円。確認したら“Produce to Checkout”のボタンを押すと、アカウント登録のページが出るのでNew Customerの項目にCreating an account only takes a few moments and allows you to store your details for future visits. If this is your first visit click here とあるので、click hereで登録する。e-mailのアドレスとpasswordでログインするようになるので登録は必須となる。その後､クレジットカード番号などを登録し、作業を完了してメールを待つ。すると、購入確認メールが送られてくる。ここで筆者はミスを犯した。CLAWSの購入の条件がacademic に関係する立場の者であることを忘れていた。Director of UCREL and Reader in Natural Language ProcessingのDr. Paul Raysonから“Thanks for the purchase of CLAWS. Please provide your academic email address for me to send the software to.”というメールを受け取った。つまり、一般のメールでは学術に関係している人かどうかわからないので、大学のメールを送って欲しいとのことだった。***@toyo.jp の自分のアドレスを記すが、大学特有のドメインである“ac.jp”ではないため、一応大学のホームページのアドレスも書いて送った。するとメールでダウンロード用ページとパスワードの指示が来た。

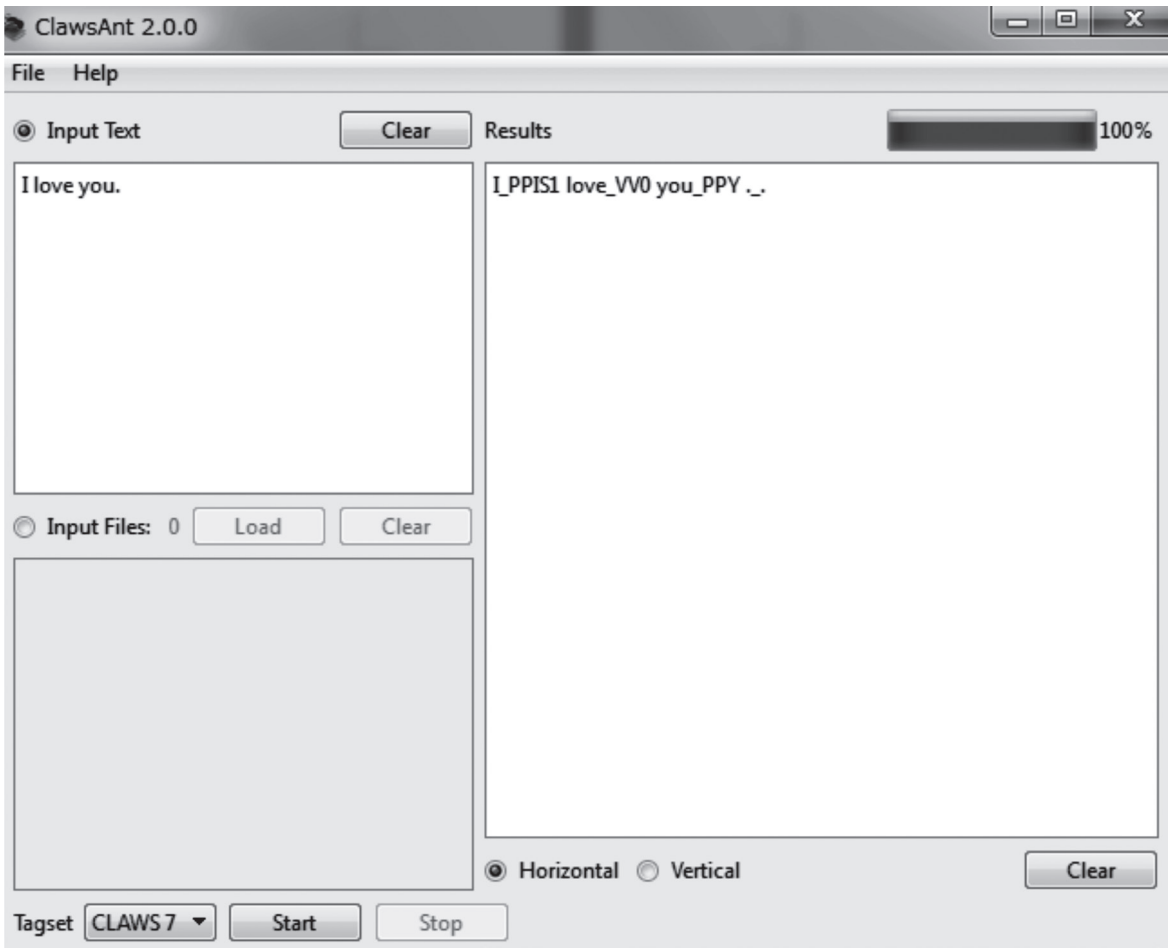
zipファイルをダウンロードして解凍すると、Windows版の他に、Mac OSX版とLinux版のファイルも入っている。Windows版はスタンドアローンなので解凍するだけでインストールは終了。本来はJAVAで動いているため設定をしなければいけない。しかし、AntConcを開発しているAntLabでCLAWSのWindows GUIを作成しくれていた。そのプログラムもClawsAnt Betaとして同梱してある。無理にJAVAの設定をしなくてもこれだけで十分に動く。ただし、所有しているWindows 8.1のコンピュータでは動かない機種もあったので、その場合はJAVA版のソフトウェアを使うしかない。ClawsAntのプログラムはCLAWSのディレクトリと同じディレクトリに置かなければならないので、移動してはいけない。

4. CLAWS の GUI と結果

ClawsAnt 2.0のGUIは単純で、機能は二つ。Input Textがテキストボックスに入れたテキストに対してtagを付ける機能で、Input FilesがTextファイルを指定して、そこに書かれているテキストにtagを付ける機能である。試験的にInput Textでボックスに“I love you.”と入れてみる。表示モードはHorizontalとVerticalがあるが、基本的にはHorizontalを使う。Tagsetを選択できるが、CLAWS7しかないのでそのままにしてStartボタンを押すと、隣のResultsボックスにtag付きのテキストが現れる。これはコピー＆ペーストが可能だ。

(1) I_PPIS1 love_VV0 you_PPY ._.

アンダーバーとスペースで挟まれた部分が文法tagとなる。ピリオドなどの句読記号にも



付いている。

次にInput Filesを検証してみる。Project GutenbergからダウンロードしたWilla Catherの“MY ANTONIA”を使用した。テキストサイズは432KB。ペーパーバック版で371ページ。Loadからディレクトリを指定し、読み込むとファイル名がテキストボックス内に現れる。これも、Startボタンを押すだけである。筆者のPCを使って約20秒で全ての単語に文法タグを付けられた。注意する点は、文字コードを必ずUTF8にすることである。日本製のテキストエディターはShift-JISなどの日本語用の文字コードがデフォルトであることが多い。これを文字コード指定でUTF8で保存し直してからCLAWSを使うことが必須となる。tagを付けた結果は同じディレクトリに、ファイル名に“_tagged”という文字が加えられて保存される。antonia-utf8.txtが元のファイルであれば、antonia-utf8_tagged.txtというファイル名で保存される。tagが付けられたファイルの大きさは779KB。倍までは至らない。

(2)

My Antonia, by Willa Cather

INTRODUCTION

LAST summer I happened to be crossing the plains of Iowa in a season of intense heat, and it was my good fortune to have for a traveling companion James Quayle Burden--Jim Burden, as we still call him in the West. He and I are

(3)

My_APPGE Antonia_NP1 ,_, by_II Willa_NP1 Cather_NP1 INTRODUCTION_NN1 LAST_MD summer_NNT1 I_PPIS1 happened_VVD to_TO be_VBI crossing_VVG the_AT plains_NN2 of_IO Iowa_NP1 in_II a_AT1 season_NNT1 of_IO intense_JJ heat_NN1 ,_, and_CC it_PPH1 was_VBDZ my_APPGE good_JJ fortune_NN1 to_TO have_VHI for_IF a_AT1 traveling_JJ@ companion_NN1 James_NP1 Quayle_NP1 Burden--Jim_NP1 Burden_NN1 ,_, as_CSA we_PPIS2 still_RR call_VV0 him_PPHO1 in_II the_AT West_ND1 ._.
He_PPHS1 and_CC I_PPIS1 are_VBR

(2) が元のテキストで、(3) が文法tagを付けたものである。タイトルと作者、章のタイトルと本文1 sentence（ピリオドまで）と4単語を記した。見てわかるように、tag付けされた後、改行は外される。そしてピリオドで改行される。Question mark“?”などの終了記号でも改行される。つまり、ひとつのセンテンスをデータのひとつの単位として認識するデータベースを作成していることになる。

5. 名詞の tagset

Windows用のCLAWSはCLAWS5（C5）であると、Lancaster大学のホームページで紹介されていたが、実際はCLAWS7（C7）であった。よってtagsetの数140となる。C5のtagset数が61であるのに対し、倍以上である。Brill’s Taggerはpunctuationsを含めて48、Tree taggerは58なので、その多さがうかがえる。昨年研究したBritish National Corpusのタグ付けはC5なのでかなり進化していることになる。本稿では紙面の関係上、名詞と動詞に限って調査を行った。

まず、名詞を見てみよう。

NN　common noun, neutral for number（e.g. sheep, cod, headquarters）
NN1　singular common noun（e.g. book, girl）
NN2　plural common noun（e.g. books, girls）
NNA　following noun of title（e.g. M.A.）

NNB preceding noun of title (e.g. Mr., Prof.)
NNL1 singular locative noun (e.g. Island, Street)
NNL2 plural locative noun (e.g. Islands, Streets)
NNO numeral noun, neutral for number (e.g. dozen, hundred)
NNO2 numeral noun, plural (e.g. hundreds, thousands)
NNT1 temporal noun, singular (e.g. day, week, year)
NNT2 temporal noun, plural (e.g. days, weeks, years)
NNU unit of measurement, neutral for number (e.g. in, cc)
NNU1 singular unit of measurement (e.g. inch, centimetre)
NNU2 plural unit of measurement (e.g. ins., feet)
NP proper noun, neutral for number (e.g. IBM, Andes)
NP1 singular proper noun (e.g. London, Jane, Frederick)
NP2 plural proper noun (e.g. Browns, Reagans, Koreas)
NPD1 singular weekday noun (e.g. Sunday)
NPD2 plural weekday noun (e.g. Sundays)
NPM1 singular month noun (e.g. October)
NPM2 plural month noun (e.g. Octobers)
PN indefinite pronoun, neutral for number (none)
PN1 indefinite pronoun, singular (e.g. anyone, everything, nobody, one)
PNQO objective wh-pronoun (whom)
PNQS subjective wh-pronoun (who)
PNQV wh-ever pronoun (whoever)
PNX1 reflexive indefinite pronoun (oneself)
PPGE nominal possessive personal pronoun (e.g. mine, yours)
PPH1 3rd person sing. neuter personal pronoun (it)
PPHO1 3rd person sing. objective personal pronoun (him, her)
PPHO2 3rd person plural objective personal pronoun (them)
PPHS1 3rd person sing. subjective personal pronoun (he, she)
PPHS2 3rd person plural subjective personal pronoun (they)
PPIO1 1st person sing. objective personal pronoun (me)
PPIO2 1st person plural objective personal pronoun (us)
PPIS1 1st person sing. subjective personal pronoun (I)
PPIS2 1st person plural subjective personal pronoun (we)
PPX1 singular reflexive personal pronoun (e.g. yourself, itself)

PPX2　plural reflexive personal pronoun（e.g. yourselves, themselves）
PPY　2nd person personal pronoun（you）

NNが名詞で、PNが代名詞で、40種類ある。BNCのC5の名詞と代名詞を合わせても8つしかない。分類が細かくなっていることがわかる。以下はC5の名詞のtagsetである。

NN0　Common noun, neutral for number（e.g. aircraft, data, committee）
NN1　Singular common noun（e.g. pencil, goose, time, revelation）
NN2　Plural common noun（e.g. pencils, geese, times, revelations）
NP0　Proper noun（e.g. London, Michael, Mars, IBM）
PNI　Indefinite pronoun（e.g. none, everything, one [as pronoun], nobody）
PNP　Personal pronoun（e.g. I, you, them, ours）
PNQ　Wh-pronoun（e.g. who, whoever, whom）
PNX　Reflexive pronoun（e.g. myself, yourself, itself, ourselves）

C7のNNのneutralの項目が特徴的だ。単複同形や複数表現で単数を表す名詞の範疇を別に作っている。sheepを単数か複数かは、現在形なら区別できるはずである。

(2) The sheep jumps.
(3) The sheep jump.

(2) が単数で、(3) が複数なのは三単現のsの有無で判断できるはずである。これをC7にかけてみる。

(4) The_AT sheep_NN jumps_VVZ
(5) The_AT sheep_NN jump_VV0

どちらのsheepもNNという範疇に入れているだけで、文法的に判断しているわけではない。BNCのC5でもNN0のみの結果であった。つまり、CLAWSはneutral for numberの単語に対しては文法的に思考しているわけではなく、機械的にtagを付与しているだけだということがわかった。fishのときはfish_NN となるが、fishesのときはfishes_NN2と複数の意味のtagを付与する。屈折語尾が-s及び-esになっているときはNN1に対してもNNに対しても複数と判断する。

場所の固有名詞は登録してあるだろうIslandや Streetなどに反応する。世界中の島や通り

が登録してあるわけではなく、前に大文字で始まる固有名詞がある場合に反応するようだ。

(6) Victoria Island: Victoria_NP1 Island_NNL1
(7) victoria Island: victoria Island_NN1
(8) I like victoria.: I_PPIS1 like_VV0@ victoria ._.
(9) hashtag: hashtag_VV0
(10) I like this hashtag. I_PPIS1 like_VV0@ this_DD1 hashtag_NN1 ._.
(11) Defghijk Island: Defghijk_NP1 Island_NNL1

故意にvictoriaのVを小文字にして一般の名詞に似せてみた。(7) で、前の単語が固有名詞で大文字開始になっていない場合、NNL1は付与されず、大文字開始であってもNN1の一般名詞と判断している。また、victoriaのようなCLWAS内部の辞書にない単語には反応せず、tagを付けない。新しい単語にはNN1を付与するかを試してみた。selfieにはNN1付与したが、hashtagには何とVV0を付けた。(10) のように統語的力を加えるとNN1と判断する。辞書には語彙籍的に動詞として入っているものも、文の中で名詞と判断する文法解析能力は素晴らしい。一方、(11) のようにDefghijkのような意味のない創作した文字列でも先頭を大文字にすれば、どんな言葉でも固有名詞と判断するようだ。

NNO, NNO2, NNT1, NNT2, NNU, NNU1, NNU2の数字及び単位のtaggingに関してはC7は完璧である。1対1対応で処理できるためだ。辞書のデータベースに登録さえしてあれば、単純にtagを付与することができるからだ。ただし、純粋な数字は名詞として扱っておらず、数詞として独立させている。

MC　cardinal number, neutral for number (two, three..)
MC1　singular cardinal number (one)
MC2　plural cardinal number (e.g. sixes, sevens)
MCGE　genitive cardinal number, neutral for number (two's, 100's)
MCMC　hyphenated number (40-50, 1770-1827)
MD　ordinal number (e.g. first, second, next, last)
MF　fraction, neutral for number (e.g. quarters, two-thirds)

(12) He is three years old.: He_PPHS1 is_VBZ three_MC years_NNT2 old_JJ ._.
(13) He is 3 years old.: He_PPHS1 is_VBZ 3_MC years_NNT2 old_JJ ._.
(14) He is a three-year-old boy.: He_PPHS1 is_VBZ a_AT1 three-year-old_JJ boy_NN1 ._.
(15) I know the one. I_PPIS1 know_VV0 the_AT one_PN1 ._.

(16) The two men and one woman were arrested.: The_AT two_MC men_NN2 and_CC one_MC1 woman_NN1 were_VBDR arrested_VVN ._.

threeと3は、両方ともMCに入る。a three-year-old boyとハイフンで形容詞にした場合、一つの単語として処理し、JJ（general adjective）として判断してくれる。oneは、人を表す場合はPN1で、数字を表す場合はMCと区別できている。定冠詞で判断しているようだ。

NP, NP1, NP2, の固有名詞は大文字開始で判断できる。しかし、単語ごとに判断しているため、以下のようなミスも起きる。

(17) the United Nations: the_AT United_JJ Nations_NN2
(18) the Andes is so high.: the_AT Andes_NP2 is_VBZ so_RG high_JJ ._.

the United Nationsは単数扱いだが、Unitedを形容詞として判断したため、Nationsを複数とした。the United Nationsという連語にtagを付けているわけではないので、ミスとは言い難いが、連語の固有名詞を一つの語として扱うことができないので仕方のないことか。また、tagsetの表ではNP　proper noun, neutral for number（e.g. IBM, Andes）と書いてあるのにもかかわらず、AndesはNP2と複数と判断してしまっている。そもそもNPのtagが存在しないように思える。人がこのあたりを判断するとき、ロジックではなく経験、知識として単数として判断するので、プログラムでは難しいのであろう。NPD1, NPD2, NPM1, NPM2の、月、週の固有名詞の判断は完璧である。

PN, PN1, PNQO, PNQS, PNQV, PNX1, PPGE, PPH1, PPHO2, PPHS1, PPHS2, PPIO1, PPIO2, PPIS1, PPIS2, PPX1, PPX2, PPYの代名詞も単語が決まれば品詞が決定するものが多いのでわかりやすいが、関係代名詞だけは疑問詞とぶつかることがある。

(19) I know the man who stole the bicycle.: I_PPIS1 know_VV0 the_AT man_NN1 who_PNQS stole_VVD the_AT bicycle_NN1 ._.
(20) Who is he?: Who_PNQS is_VBZ he_PPHS1 ?_?
(21) I don’t know what to do.: I_PPIS1 do_VD0 nt_XX know_VVI what_DDQ to_TO do_VDI ._.
(22) All that you love will be carried away.: All_DB that_CST you_PPY love_VV0 will_VM be_VBI carried_VVN away_RL ._.

PNQSはsubjective wh-pronounとだけ説明してあるので、疑問詞か関係代名詞かまで言及していない。よって（19）と（20）のwhoのtagは同じPNQSになってしまう。(21）のwhat

はDDQ はwh-determiner（which, what）と、wh-形の限定詞というカテゴリーに入れている。determinerというカテゴリーに全てまとめてしまうことには疑問が残る。(22）のthatは関係代名詞だが、CST（that as conjunction）というtagに入る。C7のtagsetに関係代名詞の項目はない。

PP-から始まるtagは全て代名詞である。人称や格によって下位区分されている。

(23) I love her.: I_PPIS1 love_VV0 her_PPHO1 ._.
(24) Her name is Jane.: Her_APPGE name_NN1 is_VBZ Jane_NP1 ._.
(25) You know me.: You_PPY know_VV0 me_PPIO1 ._.
(26) I know you.: I_PPIS1 know_VV0 you_PPY ._.

herは所有格と目的格が同じだが、文法的に区別している。APPGE は、possessive pronoun, pre-nominal（e.g. my, your, our）で所有格だが、構造上形容詞となるので、最初のアルファベットがAになっている。形容詞のカテゴリーはJ-になるので、APPGEは名詞でも形容詞でもないことを意味している。youのPPYは主格と目的格の区別をしていない。herの区別ができるのに、youを区別したtagを作らないのは何故か。同じ単語でも区別したいからこそのtagであるにもかかわらず、そういうものが多い。この点がもう少し進歩して欲しい個所である。

6. 動詞の tagset

次に、動詞について調査する。

VB0　be, base form（finite i.e. imperative, subjunctive）
VBDR　were
VBDZ　was
VBG　being
VBI　be, infinitive（To be or not... It will be ..）
VBM　am
VBN　been
VBR　are
VBZ　is
VD0　do, base form（finite）
VDD　did
VDG　doing

VDI　do, infinitive（I may do... To do...）
VDN　done
VDZ　does
VH0　have, base form（finite）
VHD　had（past tense）
VHG　having
VHI　have, infinitive
VHN　had（past participle）
VHZ　has
VM　modal auxiliary（can, will, would, etc.）
VMK　modal catenative（ought, used）
VV0　base form of lexical verb（e.g. give, work）
VVD　past tense of lexical verb（e.g. gave, worked）
VVG　-ing participle of lexical verb（e.g. giving, working）
VVGK　-ing participle catenative（going in be going to）
VVI　infinitive（e.g. to give... It will work...）
VVN　past participle of lexical verb（e.g. given, worked）
VVNK　past participle catenative（e.g. bound in be bound to）
VVZ　-s form of lexical verb（e.g. gives, works）

C7の動詞のtagsetの数は31。C5の25種類と比較すると、増えてはいるが名詞のような劇的な増加ではない。C5の時点である程度完成していたとも考えられるが、プログラムとしてこれ以上の精度は難しいというのが正直なところであろう。例えば、C5でVBB（The present tense forms of the verb BE）としたものを、VBM, VBR, VBZ（am, are, is）の三つに分けただけのことである。この区別に意味がないとは言わない。例えばareとwereを検索したいときには、_VB*Rでヒットさせることができる。しかし、これはtagを使わなくても、“are|were”で代用可能だ。tagsetはtagがなければ調べられないものに対して付与するべきである。C7はC5に比べて無駄なtagが増えたように思える。

では、be動詞から検証していこう。

VB0, VBDR, VBDZ, VBG, VBI, VBM, VBN, VBR, VBZがbe動詞のtagsetである。各人称に対して現在と過去に区別され、さらに準動詞をbe動詞のカテゴリーに入れている。VBG（being）については、現在分詞なのか動名詞なのかを言及していない。

(27) Be quiet!: Be_VB0 quiet_JJ !_!

(28) They proposed that the laboratory be built.: They_PPHS2 proposed_VVD that_CST the_AT laboratory_NN1 be_VBI built_VVN ._.
(29) I want to be free.: I_PPIS1 want_VV0 to_TO be_VBI free_JJ ._.

注目すべきはVB0というtagである。VBIと区別している。VB0は命令法や仮定法の場合とあるので、両方の例文にtagを付与してみた。命令法はVB0のtagが付いたが、仮定法現在の場合はVBIを付与した。shouldの省略と考えれば原形動詞とするのもわかるが、仮定法の判断ができないというのが正直なところであろう。それでも、命令法に対応しているので、これは例文検索に役立つ。

VD0, VDD, VDG, VDI, VDN, VDZが動詞doで、VH0, VHD, VHG, VHI, VHN, VHZがhave動詞である。have動詞を独立させているところが、イギリス製のtaggerらしい。

(30) Don’t mind.: Do_VD0 nt_XX mind_VVI ._.
(31) Do you play tennis?: Do_VD0 you_PPY play_VVI tennis_NN1 ?_?
(32) I do love you.: I_PPIS1 do_VD0 love_VVI you_PPY ._.
(33) Have you this pen?: Have_VH0 you_PPY this_DD1 pen_NN1 ?_?
(34) She has gone to the man.: She_PPHS1 has_VHZ gone_VVN to_II the_AT man_NN1 ._.
(34) I have to go there.: I_PPIS1 have_VH0 to_TO go_VVI there_RL ._.

doもhaveも、動詞の現在形と助動詞の区別をしておらず、VD0/VH0である。一般動詞のdoと助動詞のdoの区別ができない。たとえば、検索で“do_VD0 *_V*I”と入力すれば強意の助動詞のdoの例文をヒットさせることは可能だが、tag単独で探すことはできない。have toも連語なので助動詞としては判断してくれない。

では、その助動詞はどう扱っているだろうか。VM　　とVMK　　が助動詞で、動詞カテゴリーの中にある。have toは助動詞として扱ってくれなかったが、他の表現ではどうだろう。

(35) I ought to go there.: I_PPIS1 ought_VMK to_TO go_VVI there_RL ._.
(36) She used to play with him.: She_PPHS1 used_VMK to_TO play_VVI with_IW him_PPHO1 ._.
(37) This metal is used to make the car.: This_DD1 metal_NN1 is_VBZ used_VVN to_TO make_VVI the_AT car_NN1 ._.
(38) You had better be home.: You_PPY had_VHD better_RRR be_VBI home_RL ._.

(39) He needn't go there.: He_PPHS1 need_VM@ n't_XX go_VVI there_RL ._.

have toは助動詞として認めなかったにもかかわらず、ought toやused toは連語として認めている。この違いはどこにあるのだろうか。have toをhave_VMK to_TOとできなかった理由は何か。(36) はused toを助動詞とし、(37) は受動態＋不定詞と、正確に判断している。ここから、have toに助動詞の意識が薄く、have動詞＋不定詞として捕えているのではないかという推測が付く。had betterも助動詞にはしていない。一方、needの助動詞用法はVMとして判断している。完了形のhaveも助動詞としては捕えていないことからも、haveはhave動詞というカテゴリーに入れたのではないだろうか。

一般動詞のtagsetはVV0, VVD, VVG, VVGK, VVI, VVN, VVNK, VVZである。be動詞との大きな違いは、原形と現在形が同じことである。よって外見上、三単現の動詞以外、命令形と現在形の区別が付かない。統語的な判断もしていない。また、進行形と動名詞の区別も出来ていないようである。

(40) My hobby is playing tennis.: My_APPGE hobby_NN1 is_VBZ playing_VVG tennis_NN1 ._.
(41) My father is playing tennis.: My_APPGE father_NN1 is_VBZ playing_VVG tennis_NN1 ._.

二つの文、ともに全く同じtagを付与している。統語的なプログラムだけではここに限界がある。My hobbyは絶対にテニスをしない。My fatherはテニスをすることではない。これらを判断させるには、意味tagをさらに付与する必要があり、さらに意味に対するアルゴリズムを作成する必要がある。tagを正確に付与できるということは、日本語に訳せるということになる。そこでネット上にある翻訳機を当たってみたが、両方の文を正確に日本語に訳せるものはなかった。つまり、現在の最新コンピュータでも、まだそこまでは至っていないということになる。

7. Taggerの未来

本稿では、Taggerそのものに焦点を当てたが、それを使ってTagを付与したコーパスをいかに使っていくかが本来の目的である。また、tagが付与できるということは、コンピュータが文章を「理解」したということに近付いている。今はまだアルゴリズムによるtaggingが中心だが、ビッグデータを使い、文法tag以外のtagを付与することも可能となるだろう。例えば色や形状、動き方などの抽象的概念のtagである。そうすれば高性能な翻訳機も作ることができるようになると思う。

参考サイト

・CLAWS 購入

http://online-payments.lancaster-university.co.uk/browse/product.asp?compid=1&modid=1&catid=194

・神戸大学石川慎一郎研究室

http://language.sakura.ne.jp/s/corpus.html

・CLAWS7 tagset

http://ucrel.lancs.ac.uk/claws7tags.html

A study of tagger

NAKURA, Hideto

For some years, I have studied about corpus, for example, Brown Corpus, Lob Corpus and British National Corpus and so on. In the meantime, I got interested in the very tags of corpus. About corpus, tags mean part-of-speech markers. Each word has the marker. Using the markers, we can investigate the construction of English. But how were tags provided on the words in corpus? Can computers analyze the structure of English? So, I got a tagger, which can assign tags to words. The name is CLAWS, which made British National Corpus a corpus with tags. So I researched CLAWS by giving this tagger various sentences. I investigated how CLAWS is correct and when it mistakes.

In this article, first of all, I explained the history of taggers. After that, I researched CLAWS tags, mainly nouns and verbs.